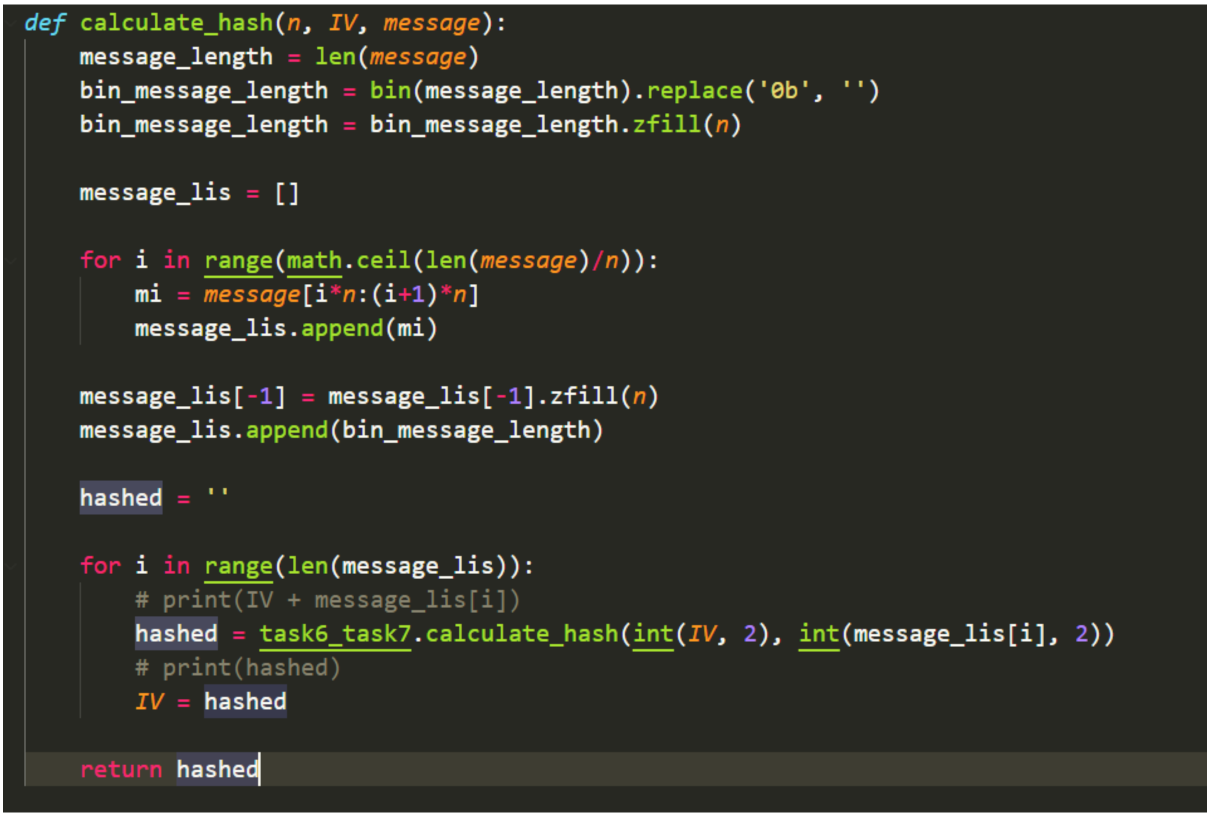
Task 7 Merkle Damsgard Transformation



```
def calculate_hash(n, IV, message):
    message_length = len(message)
    bin_message_length = bin(message_length).replace('0b', '')
    bin_message_length = bin_message_length.zfill(n)

    message_lis = []

    for i in range(math.ceil(len(message)/n)):
        mi = message[i*n:(i+1)*n]
        message_lis.append(mi)

    message_lis[-1] = message_lis[-1].zfill(n)
    message_lis.append(bin_message_length)

    hashed = ''

    for i in range(len(message_lis)):
        # print(IV + message_lis[i])
        hashed = task6_task7.calculate_hash(int(IV, 2), int(message_lis[i], 2))
        # print(hashed)
        IV = hashed

    return hashed
```

Calculate_hash()

The function takes n, IV, and message as input parameters, n here denotes the input size of fixed length hash which the function makes use of. The function breaks down the message into n bit chunks, hash is calculated on each block and the final hash is returned by the function.

Sample Input

Fixed hash Length: 16

Message: 1010101010111111111111000011011101010000011101001010101111111110000000

Sample Output

Hash Output: 0110010011100000 16